CENTRE FOR
SYSTEMS
SCIENCE

Simon Fraser University, Burnaby, B.C., Canada V5A 1S6 / (604) 291-4704

92-05405/

Fast Collision Avoidance for Manipulator Arms: A Sequential Search Strategy

Kamal Kant Gupta

CSS-IS TR 88-03

Fast Collision Avoidance for Manipulator Arms: A Sequential Search Strategy

Kamal Kant Gupta*
School of Engineering Science
Simon Fraser University, Burnaby, B.C. V5A 1S6.

December 15, 1988

Abstract

We present a novel sequential strategy to plan collision-free motions for a manipulator arm (such as PUMA 260). The basic idea behind our approach is to plan the motion of each link successively starting from the base link. Given that the motion of link i has been planned, the motion of link $i+1$ is planned by controlling the degree of freedom associated with $i+1$ link, in essence solving a 1-dimensional motion planning problem. This results in n 1-dimensional planning problems instead of one n-dimensional problem. This 1-dimensional motion planning problem is solved by using fast and efficient techniques similar to [Kant and Zucker 1986]. Thus, the algorithm is fast and efficient, and has potential for real-time implementation. The price paid, however, is that the algorithm may miss certain paths. We believe that in most reasonable cases, the algorithm will find a path. At the very least, it will form a front end to a more complete planner. Furthermore, this has the potential of incorporating some aspects of dynamics right at the planning stage, i.e., incorporating the bounds on velocity and acceleration at the planning level.

*This research was supported by an NSERC operating grant and SFU President's research grant.

1 Introduction

Motion planning for manipulator arms has been a much researched topic [Udupa 1977, Lozano-Perez 1981, 1987, Brooks 1983, Schwartz and Sharir 1982]. As generally stated in the literature, the motion planning problem is to find a path from a given initial robot configuration to a given final configuration that avoids collisions with a known set of stationary obstacles in its workspace. Many of these algorithms are based on the use of configuration space (Cspace) – the space of degrees of freedom of the robot. This permits explicit characterization of the constraints that the physical obstacles impose on the robot's motion. However, there are two major difficulties with this approach for practical manipulator arms: (i) high dimensionality of the Cspace, and (ii) complexity of the Cspace obstacles. For instance, for a six degree of freedom arm, building and searching the six dimensional Cspace, although possible [Donald 1984], is extremely cumbersome and slow. Especially with increasing emphasis on redundant arms, this problem of dimensionality is going to be worse. Many different approaches [Faverjon 1984, Gouzenes 1984, Lozano-Perez 1987, Xing 1987] have been suggested to alleviate this problem. Whether explicitly or implicitly, these approaches build and search a lower dimensional subspace of the full Cspace. [Lozano-Perez 1987] has implemented a motion planning algorithm that demonstrates impressive results, but still takes the order of a few seconds for the simple case of a two degree of freedom planar arm. Clearly, it will take much longer for a full six degree of freedom three dimensional arm.

The approach taken here is related to the one described by Lozano-Perez in that it uses discretized Cspace. However, our emphasis is on *efficiency*, and hence we suggest a radically different way of building and searching an n-dimensional Cspace – building and searching n 1-dimensional subpaces. First we give an intuitive explanation of our approach, and then discuss its pros and cons.

First, consider an interesting observation about human arm motion. Consider the motion of your arm as you reach out to pick up a book on the table – you start moving your arm toward the book. if there is an obstacle (a stack of other books) say in the way of your forearm, your upper arm motion reamins the same, but you rotate your forearm out of the way to avoid the obstacle. Similarly, if there is an obstacle say in the way of your hand, the motion of your upper arm and forearm remains the same, but you rotate your hand out of the way to avoid the obstacle. Other researchers have pointed out this sequential nature of human arm motion in a different context. For example, [Korein 1985] mentions that to reach and pick up an object, humans use the proximal joints first and the distal ones later. From motion planning perspective, the key observation is that in many human arm motions collision-avoidance is done (i) sequentially from proximal to distal, and (ii) one joint at a time.

This simple observation motivates our algorithm as follows. A simple manipulator arm (like a human arm) has an inherent ordering of links, from base link to the last link. Consider a 3 link (planar manipulator) shown in figure 1. The links have been labeled in increasing order from the base, i.e, the base link is labeled 1, the next link is labeled 2, and the last (free) link is labeled 3. Suppose, the motion of link 1 has been planned. Now, as it moves link 2 can be treated as a moving robot with one degree of freedom (pure rotation for revolute joints). Thus, its motion (rotation) can be planned to avoid the obstacles. Once, the motion of link two has been planned, now the motion of link three can be planned, and so on... ***Notice that the problem of planning in an n-dimensional Cspace (for an n-link arm) has been converted into n much simpler 1-dimensional (single degree of freedom) planning problems.*** This is shown in figure 1.

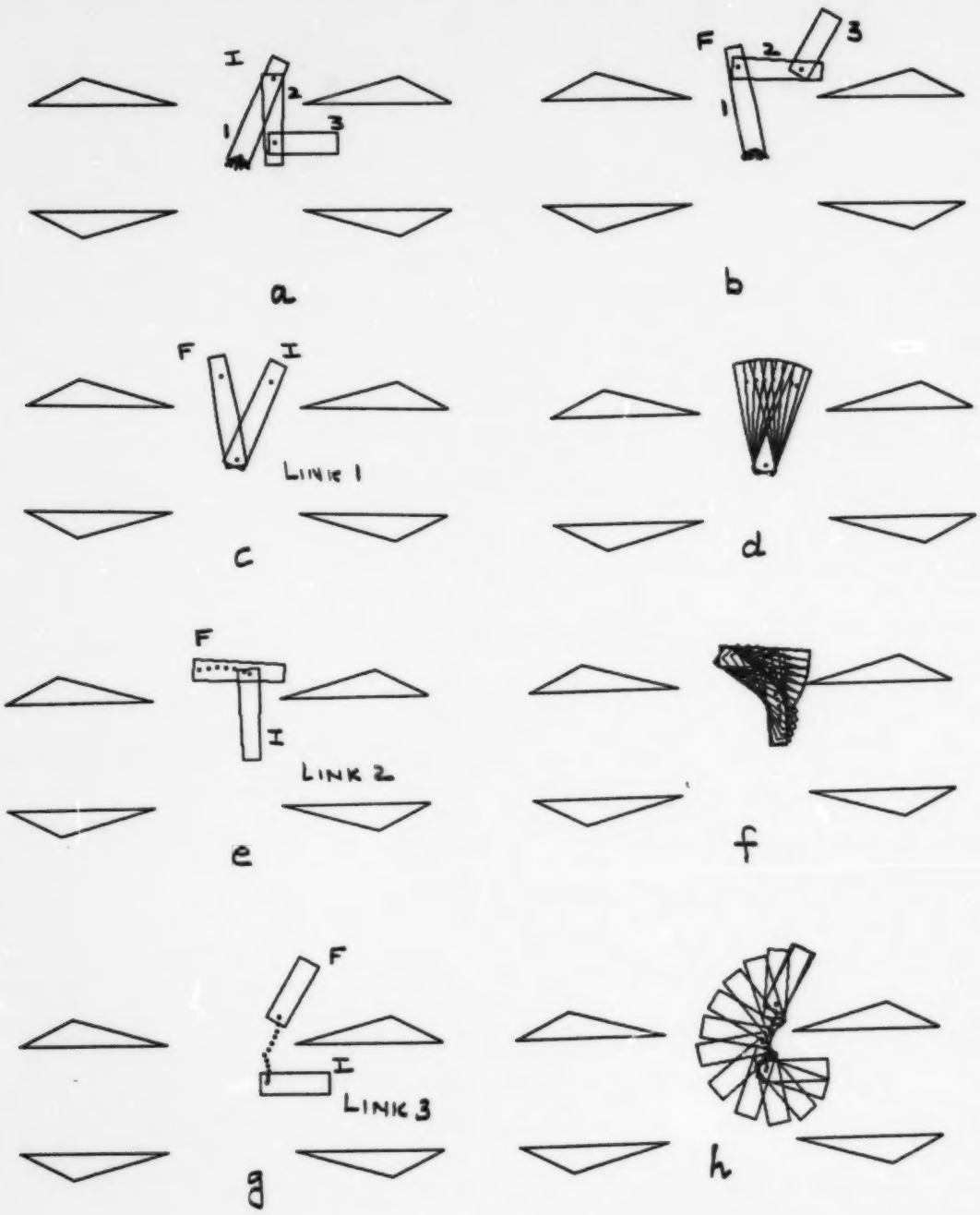

Figure 1: Decomposing the motion planning problem for an n link manipulator into into n simpler single-link problems. The initial configuration is shown in (a), and the final configuration in (b). Planning proceeds from the base link (link 1) to the last link (link 3). First the motion of link 1 is planned (c) and (d). This motion determines the motion of a reference point (point where link 2 is attached to link 1) on link 2. The angle of link 2 is now planned to avoid collisions with obstacles (e) and (f). This motion of link 2 determines the motion of a reference point (point where link 3 is attached to link 2) on link 3. The angle of link 3 is now planned (g) and (h). This process is repeated for an n times for an n-link arm, resulting in n 1-dimensional planning problems.

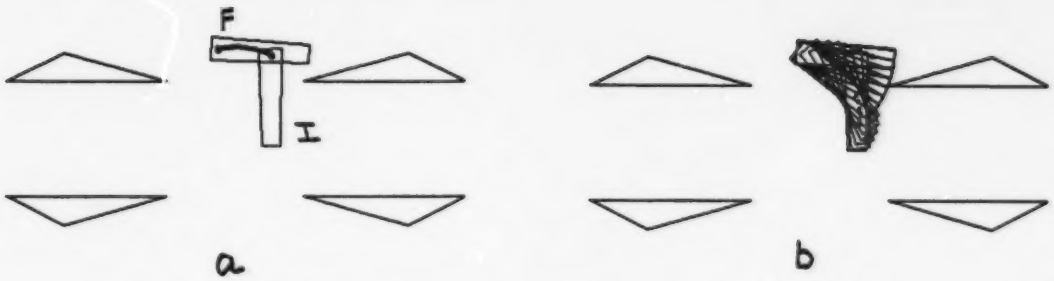

Figure 2: The single-link problem: given the path of a fixed reference point on the link, plan its angle from initial configuration I to final configuration F such that it avoids collisions with the obstacles. The path of the reference point is a circular segment. This is shown in (a). The angle of the link is planned to avoid collisions with the obstacles. The resulting motion is shown in (b).

How do we solve this one degree of freedom problem? Note that the motion of one end of the link is already known (the motion of previous links having been already planned. As the link moves along this path, the problem now is to adjust its angle (or equivalently its rotational velocity) to avoid the obstacles in its way. See figure 2. This problem is akin to the one solved in [Kant and Zucker 1986, 1988].

We parameterize the path of the link endpoint by parameter t. One interpretation of this parameter is time, although it is not necessarily so. The physical obstacles give rise to constraints on the angle θ of the link as a function of t. These constraints are represented as regions in $t \times \theta$ space. A path in this θ space represents a collision-free motion for the link. However, there is an additional constraint on feasible paths in $t \times \theta$ space – they are monotonic functions of parameter t. This constrained search problem has been extensively explored in [Kant and Zucker 1986, 1988]. Thus, those search techniques are then used to find a collision-free motion for the link. Moreover, we can incorporate bounds on angular velocity (by methods similar to [Kant and Zucker 1986, 1988] and angular acceleration [O'Dunlaing 1987, Gupta and Yalpani 1989]. However, in this paper we are mainly concerned with efficiency, and will leave the issue of the dynamic constraints with a mention that they can be incorporated in the planning process by applying techniques developed in the above references.

An example for a 2-D four link arm among polygonal obstacles is shown in figure 3. Although our implementations are for a 2-D case, note that it is straight forward to extend it to a 3-D world.

Our algorithm can be implemented on-line, since, the motion of only one link at a time is being planned. This is a main advantage of the algorithm. A main question to be answered is under what conditions this sequential planning will fail, and how often such conditions will occur? The first thought that all situations where the motion of more than one link has to be planned simultaneously will be missed, is not true. Because, although the planning is sequential, the planning of a distal link depends upon all the previous proximal links. However, if along the path (determined by the

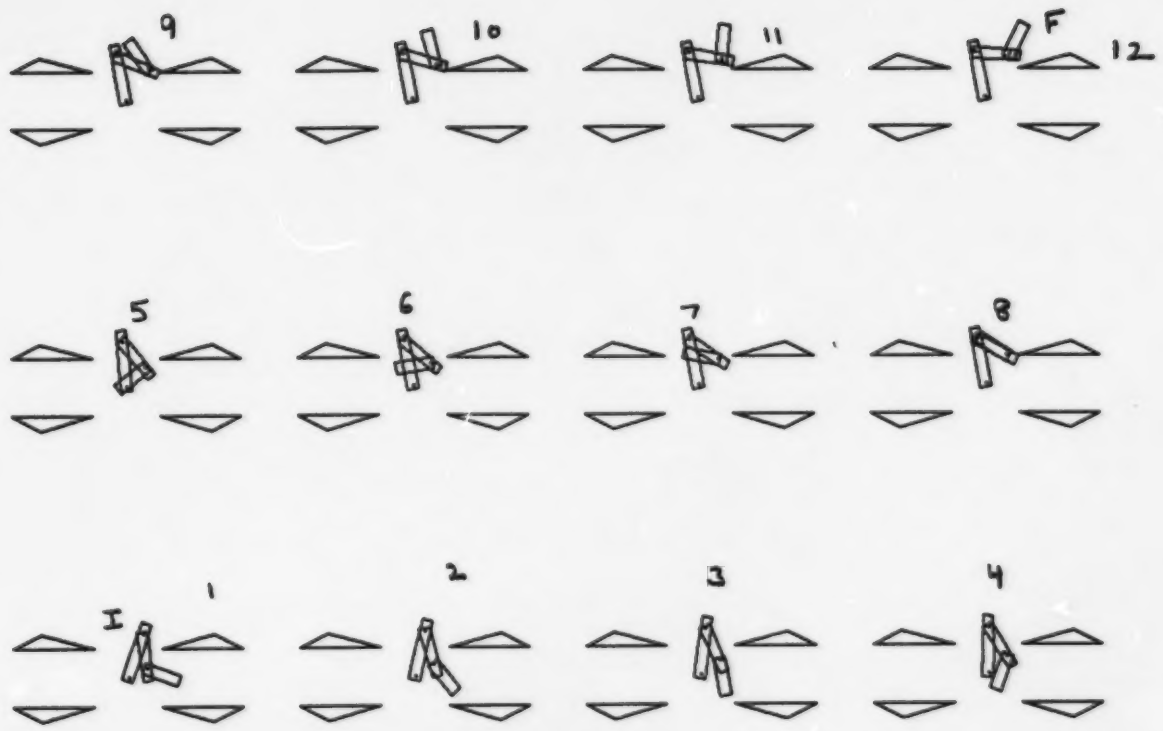

Figure 3: An example of our algorithm for a three-link planar manipulator arm. The sequence goes from left to right, and bottom to top. The start configuration is at bottom left, and the final configuration at top right.

Figure 4: Various parameters for a given arm.

previous $1, \ldots i$ links), no motion exists for $i+1^{th}$ link, clearly our sequential strategy will fail. In this case, the motion of previous links will have to be replanned. However, to decide how likely is such a situation to occur in real world, will require more experimentation. At the very least, our search strategy will form a front end to a more complete planner in identifying regions where more complete planning is needed.

In the next few sections, we present our approach in detail.

2 The Approach

Let the manipulator joint parameters be represented by q_i, the corresponding links by l_i, $i = 1, 2, \ldots, n$, l_1 being the most proximal link, and l_n the most distal link as shown in figure 4. Let $\mathbf{r_i}$ be the ref. vertex for link l_i. The base of the first link is assumed to be fixed at $\mathbf{r_1}$. Let $\mathbf{r_i}(t)$ denote the trajectory of the reference vertex $\mathbf{r_i}$. The initial and goal configurations of the arm are given as q_i^s, and q_i^g, $i = 1, n$. The algorithm is as follows:

1. $i = 1$.
2. $\mathbf{r_1}(t) = \mathbf{r_1}$
3. While $(i < n)$
   (a) along $\mathbf{r_i}(t)$, compute a collision-free trajectory for the i^{th} joint, from q_i^s to q_i^g
   (b) given $q_1(t), q_2(t), \ldots q_i(t)$, compute $\mathbf{r_{i+1}(t)}$
   (c) Increment i.

Note that the step (3b) in the while loop is forward kinematics, and is computationally straightforward [Paul 1982]. The key step in our algorithm is that of planning the motion of a single link

given that one end of the link is moving along a given path (the path may be a single point as in the case of the very first link, whose base is fixed). This is what we solve in the next section.

3 Planning one degree of freedom along a given path

Many different approaches may be suggested to deal with this problem. One approach would be to analytically characterize the constraints in the $t \times \theta$ space (see [Schwartz and Sharir 1981, 1983] for moving a rod among polyhedral obstacles). Another approach is to discretize the joint angles. Approaches based on discretization lend themselves to ready machine implementation. In this paper, our approach is also based on discretization.

Assume that the motion of the reference vertex of link l_i is known, i.e., $\mathbf{r}_1(t)$ is known. Discretize $\mathbf{r}_1(t)$ at a certain resolution, $t = kT$. We shall denote the discrete trajectory with $\mathbf{r}_1(k)$. For each $\mathbf{r}_1(k)$, the legal ranges of q_i can be easily determined as in [Faverjon 1984, Lozano-Perez 1987]. We briefly discuss it below.

3.1 Computing Forbidden Ranges

For a given position of the link reference vertex, the range of forbidden values of q_i will be bounded by angles where link l_i is just touching an obstacle. For polygonal obstacles, these critical values occur when (i) a vertex of the link is touching an edge of an obstacle (vertex-edge contact), and (ii) an edge of the link is touching a vertex of an obstacle (edge-vertex).

Consider the vertex-edge contact first. Since the link is constrained to rotate about its joint, the link vertices trace out circular paths. The intersection of these paths with the obstacle edges determine some of the critical values of q_i. The edge-vertex contact is similar, except that the edge is a link edge and the vertex belongs to the obstacle. Thus the geometric computations in determining these critical values essentially involve intersecting circles with line segments. The details are given in the [Lozano-Perez 1987]. The angle of contact is given by

$$q_i = cos^{-1}(\frac{-d}{\|v\|}) + \phi.$$

The situation is illustrated in figure 5.

For type A contact, the only difference is that the edge is a link edge and the vertex an obstacle vertex. The contact angle is given by

$$q_i = cos^{-1}(\frac{-d}{\|v\|}) - \phi.$$

Note that these just give potential contact angles. Three further conditions must be satisfied: (i) the intersection point must lie inside the edge, (ii) the edges defining the contact vertex must lie outside the obstacle, and (iii) there must not be other contacts that prevent this contact from occurring. The first constraint is easily checked by computing the co-ordinates of the the intersection point and the position of edge endpoints, given the computed values of q_k. The second constraint can be checked by

$$sgn(\mathbf{n} \cdot \mathbf{e}_1) \geq 0 \quad \text{and} \quad sgn(\mathbf{n} \cdot \mathbf{e}_2) \geq 0.$$

The third constraint requires examining all other contacts with an obstacle that satisfy the first two constraints. For each contact angle, we determine whether the collision occurs for values

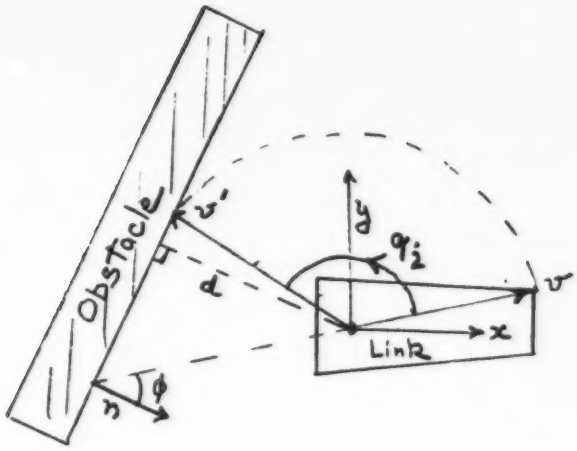

Figure 5: An illustration for terms used in deriving the contact angles.

greater than or less than the contact angle. This is easily determined by evaluating the derivative of distance d with respect to q_i. The contact angles together with the directions are then merged to form the ranges of forbidden values for q_i.

An example is shown in figure 6. The path for the reference vertex $\mathbf{r_i}$ is a straight line path. The path has been discretized at certain resolution, and at each discrete point, the forbidden ranges have been determined.

Having determined these ranges, we further group them in polygonal regions. The basic idea here is to use the structure in the Cspace obstacles to reduce the search time.

3.2 Obstacle Representation

The basic idea is to group contiguous and overlapping forbidden slices in one obstacle. We suggest a very conservative algorithm here. Note that more elaborate schemes that do not waste as much free-space can be easily devised. Our aim here is just to illustrate a simple one that conveys the idea of grouping slices into obstacles. The algorithm is as follows: Each forbidden range in the first slice starts a new obstacle. We keep track of the maximum and the minimum values of all the ranges in the obstacle. We shall call it the bounding range. As each slice is considered, the range overlapping the bounding range of a given obstacle is added to that obstacle. If more than one range overlaps a bounding range, these ranges are merged into a single range and added to that obstacle. An obstacle is terminated when no range in a slice overlaps the bounding range of the obstacle. Thus, one may get overlapping obstacles, and moreover, much of the free-space may be treated as an obstacle.

The next step is to group the slices belonging to one obstacle into bounding polygons. Many elaborate schemes from computational geometry may be used to get these bounding polygonal regions. We have used bounding convex hulls. The number of sides of the resulting polygon may still be very large, and is further reduced to a small number, in our case less than or equal to 6.

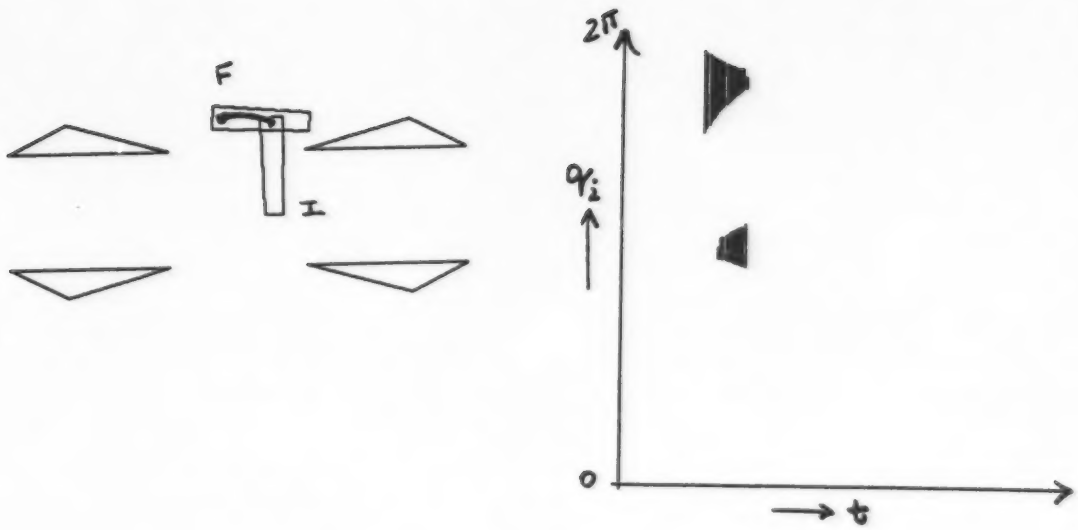

Figure 6: The path for the reference vertex $\mathbf{r}_i$ is a circular line segment, parameterized by t. It is further discretized at certain resolution $t = kT$, and at each discrete point, the forbidden ranges of q_i are shown.

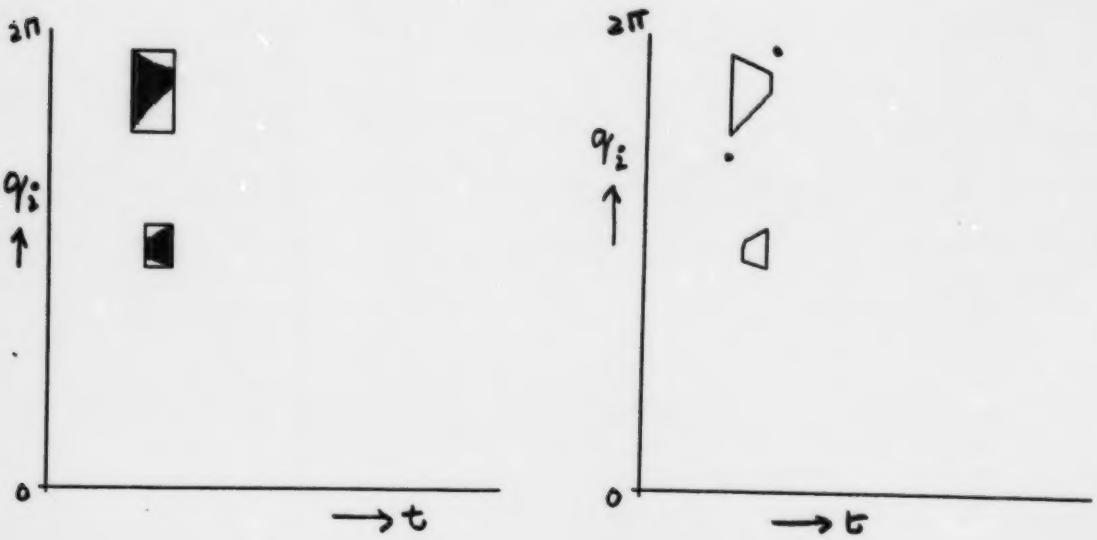

Figure 7: Grouping forbidden ranges into polygons. This involves two steps. First, the contiguous ranges are grouped as belonging to the same bounding box as shown in (a). Next, a convex hull of the ranges belonging to the same bounding box is obtained, shown in (b). A further refinement of the convex hull gives a polygon of sides less than or equal to 6.

This is simply achieved by extending the two sides to the point where they intersect. Different steps involved in the grouping of individual slices into polygonal obstacles are shown in figure 7.

3.3 Constrained Visibility Graph Search

Having characterized the forbidden regions in $t \times \theta$ space, we may now use 2-dimensional path planning approaches to search for a collision-free motion. However, there are additional constraints, since, one of the dimensions is time. 2-D search techniques with time as one of the dimensions have been extensively explored [Kant and Zucker 1986, 1988]. The two main approaches are: (i) visibility graph based [Kant and Zucker 1986], and (ii) free-space based [Kant and Zucker 88]. We use the visibility graph based approach to search the $t \times \theta$ space. We briefly mention it here, for a detailed discussion, please see the above references.

The time monotonicity constraint implies that any physically realizable mapping has to be a strictly monotonically increasing function of time t. This constraint makes the visibility graph in $t \times \theta$ space, a directed graph, since, a point (t_j, θ_j) is accessible from another point (t_k, θ_k), only if $t_k < t_j$).

More formally, we can state this as follows: Let I be the start position and F the final position in $t \times \theta$ space. Let O be the set of polygonal obstacles in $t \times \theta$ space. Let V be the set of vertices

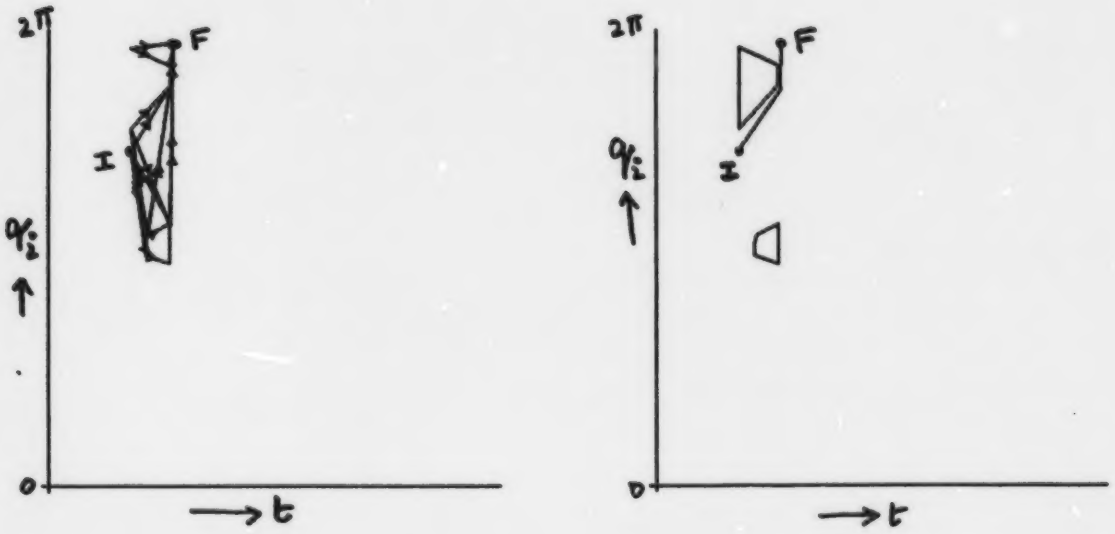

Figure 8: The directed visibility graph in $t \times \theta$ space is shown in (a). Only the edges that correspond to a positive change in parameter t are valid. The rest are pruned out. A path search from the start node I to the goal node F gives a collision-free motion. Such a path is shown in (b).

of these obstacles. The node set $N = \{I, F\} \cup V$. To determine the edge set E, we first define a visibility function l for a node pair (n_i, n_j). Let the coordinates of node $n_i = (t_i, \theta_i)$ and those of node $n_j = (t_j, \theta_j)$. Then $l(n_i, n_j)$ is TRUE if:

1. $\overline{n_i n_j} \cap O = \emptyset$,
2. $t_i < t_j$.

The edge set E is defined as the set of all edges (n_i, n_j) such that $l(n_i, n_j)$ is TRUE. An example of such a graph DVG is shown in figure 8.

Note that pruning does not loose any feasible trajectories. We state the theorem below, for a proof see [Kant and Zucker 1986].

Theorem 1 *The directed Vgraph DVG is complete. If a graph path does not exist, it implies that a collision-free trajectory from I to F does not exist.*

This directed visibility graph, DVG may then be searched for a collision-free trajectory for q_i using any of the standard search techniques. An example is shown in figure8.

4 The Complete Algorithm

Given that we have solved the problem for a single link, we can now state the full algorithm.

Find the range of legal values of q_1.
Find a collision-free trajectory for link 1, $q_1(t)$.
if (success)
  Discretize $q_1(t)$ to get $q_1(kT) \stackrel{\text{def}}{=} q_1(k)$, $k = 1, \ldots, M$.
  $i = 2$.
  while $(i \leq n)$
    for $k = 1, M$
      Given $q_1(k), q_2(k), \ldots q_{i-1}(k)$, compute $\mathbf{r}_i(k)$.
      Compute the forbidden ranges (slices) of $q_i(k)$ for each $\mathbf{r}_i(k)$.
    end for
    Group $q_i(k)$, $k = 1, M$ slices into regions.
    Group regions into polygonal obstacles.
    Build visibility graph.
    Prune edges to get a directed visibility graph.
    Search the graph to get a collision-free path for q_i, $q_i(t)$.
    if (success)
      Discretize $q_i(t)$, i.e., compute $q_i(kT)$, $k = 1, M$ along the collision-free path.
      Increment i.
    else return failure
  end while
  return success
else
  return failure

This algorithm, if successful, will give a collision-free path for the given manipulator. An example is shown for a three link planar arm in figure 3. This motion was obtained by solving three 1-dimensional problems – the $t \times \theta$ space obstacles, and the corresponding paths for each sub-problem being shown in figure 9. Our implementations are for a 2-D n-link arm. However, the extension to a 3-D arm is straightforward. Furthermore, note that the complexity of the search increases linearly with the number of degrees of freedom.

5 Conclusions

In this paper, we have suggested and implemented a strategy that results in fast and efficient planning of collision-free motions for a manipulator arm. This strategy has potential for real-time implementation on a full six degree of freedom arm. The essence of the strategy lies in building the Cspace and searching it one link at a time, starting from the base link. For an n-link arm, this means that rather than searching and building a full n-dimensional Cspace, we build and search n 1-dimensional subspaces. Of course, the price paid is that the search is not complete. But we believe that in most of the cases this strategy will be successful. Furthermore, this has the potential

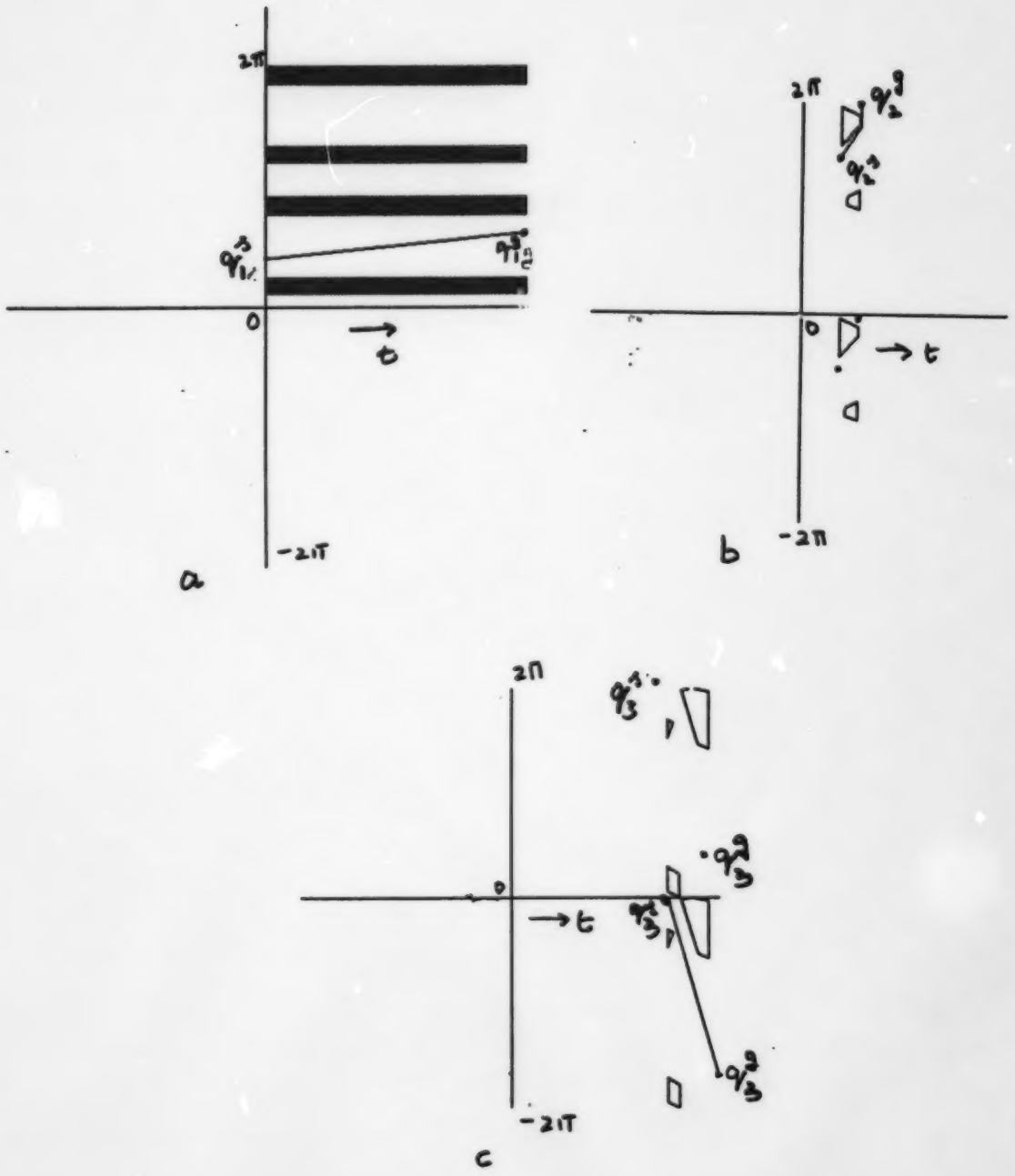

Figure 9: The motion in figure 3 was obtained as above. The $t \times \theta$ space obstacle and the corresponding paths are shown for each link – for link 1 in (a), for link 2 in (b), and for link 3 in (c). The range for angle (the vertical axis in each figure) is from -2π to 2π to explicitly show that the angle wraps around modulo 2π. See figure 1 for the corresponding motion of each link.

of incorporating some aspects of dynamics right at the planning stage, i.e., of incorporating the bounds on velocity and acceleration at the planning level.

In cases where this strategy fails to find a collision-free path, it may be generalized to higher dimensional sub-spaces. For instance, one possible approach would be to consider first two links together, i.e., solve a 2-degree of freedom problem. Given that the motion of first two links is planned, along this path, the motion for successive links may be planned. However, beyond this, the approach may loose its major advantage: speed. In fact, in extremely cluttered situations, one may have to search the full n-dimensional Cspace. But again, we believe that such situations will occur in very cluttered situations. In such situations, this strategy will form a front end to a more complete planner.

References

[1] Donald, B. 1984. Motion Planning with six degrees of freedom. MIT AI Tech. Rep. 791. 504-512

[2] Faverjon, B. 1984. Obstacle avoidance using an octree in the configuration space of a manipulator. *IEEE Conference on Robotics and Automation.* pp. 504-512

[3] Gouzenes, Laurent. 1984. Strategies for solving collision-free trajectories for mobile and manipulator robots. *Int. Journal of Robotics Research.* 3(4). MIT Press.

[4] Gupta, Kamal Kant, and Yalpani, Kuros. 1989. Motion planning with Velocity and Acceleration Constraints. To appear. Technical Report. Centre for Systems Science, Simon Fraser University.

[5] Korein, J. 1985. *A geometric investigation of reach.* MIT Press.

[6] Kant, Kamal, and Zucker, S.W. 1986. Toward efficient trajectory planning: the path-velocity decomposition. *International Journal of Robotics Research.* 5(3).

[7] Kant, Kamal, and Zucker, S.W. 1988. Planning smooth collision-free trajectories: path, velocity, and splines in free-space. *International Journal of Robotics and Automation.* 2(3).

[8] Lozano-Perez, T. June 1987. A simple motion planning algorithm for general robot manipulators. *IEEE Journal of Robotics and Automation.* 3(3).

[9] O'Dunlaing, Colm. 1987. Motion Planning with Inertial Constraints. *Algorithmica.* 2.

[10] Paul, Richard, T. June 1987. Robot Manipulators: Mathematics, programming, and control. MIT Press.

[11] Schwartz, Jacob T. and Sharir, Micha. October 1981. On the 'Piano Movers' problem I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. Report No. 39. Dept. of Comp. Sc., Courant Institute of Math. Sc., New York.

[12] Schwartz, Jacob T. and Sharir, Micha. October 1982. On the 'Piano Movers' problem II. General techniques for computing topological properties of real algebraic manifolds. Report No. 41. Dept. of Comp. Sc., Courant Institute of Math. Sc., New York.

[13] Schwartz, Jacob T. and Sharir, Micha. October 1983. On the 'Piano Movers' problem V. The case of a rod moving in three-dimensional space amidst polyhedral obstacles. Robotics Report No. 13. Dept. of Comp. Sc., Courant Institute of Math. Sc., New York.

[14] Udupa, S. 1977. Collision detection and avoidance in computer controlled manipulators. *5th Int. Joint Conf. on Artificial Intelligence.* Cambridge.

[15] Xing, D. M. 1988. Collision-free trajectory generation for a general robot manipulator. Tech. Report. CRIIF, Laboratoire de Robotique. Paris.

Centre for Systems Science

• Simon Fraser University established the **Centre for Systems Science** as a provincial centre for advanced study. The objective of CSS is twofold: coordinate and combine advanced systems research at SFU, and encourage application of research results within industry in order to further the technological development of British Columbia.

• CSS has assembled three research groups to streamline its efforts. The orientation of these groups is as follows: intelligent systems, microelectronics, and computer and communications systems. Because of shared research interests, this latter group merges with LCCR (**Laboratory for Computer and Communications Research**) for the publication of technical reports.

• CSS's three groups involve over 80 skilled faculty researchers principally from SFU's Schools of Engineering Science and Computing Science, but also draw substantially on Mathematics, Physics, Geography, Philosophy, Linguistics, Psychology, Communication, and Kinesiology.